Use Case: Tính học phí hoàn thành CTĐT

1. Thông tin chung

- **Tên Use Case**: Tính học phí hoàn thành CTĐT (Curriculum Completion Tuition Calculation).
- **Mô tả**: Use Case này mô tả quy trình tính toán học phí cần thiết để sinh viên hoàn thành chương trình đào tạo (CTĐT). Quy trình bao gồm tính học phí tổng thể, học phí theo học phần, học phí đã đóng và còn lại, dự báo học phí, với ưu tiên tính theo Chương trình đào tạo (Program) là chính. Ngành học (Major) chỉ dùng làm fallback khi chưa có dữ liệu học phí theo Program. Hỗ trợ nhiều phương thức tính (theo tín chỉ, cố định, theo học phần, hỗn hợp).
- **Actor**:
    - **Chính**: Sinh viên (xem học phí cá nhân), Quản lý tài chính (Finance Manager - tính toán, phê duyệt, quản lý), Quản trị viên hệ thống (Admin - thiết lập quy tắc học phí).
    - **Hỗ trợ**: Hệ thống quản lý đào tạo (tích hợp dữ liệu từ Program, Course, Invoice).
- **Mức độ ưu tiên**: Cao (là tính năng quan trọng trong hệ thống quản lý đào tạo).
- **Phạm vi**: Hệ thống quản lý học phí (finance schema) và học thuật (academic schema).
- **Pre-conditions**:
    - Sinh viên đã đăng ký CTĐT (có dữ liệu trong StudentAcademicProgress).
    - Có dữ liệu học phí theo ProgramTuition (ưu tiên) hoặc fallback MajorTuition/TuitionRule.
    - Hệ thống có dữ liệu cấu trúc CTĐT (Program, ProgramCourseMap, Course).
    - Người dùng (sinh viên/admin) đã đăng nhập và có quyền truy cập.
- **Post-conditions**:
    - Học phí được tính toán và lưu trữ (vào ProgramTuitionCalculation hoặc StudentTuitionSummary).
    - Báo cáo học phí được tạo (nếu yêu cầu).
    - Thông báo học phí được gửi cho sinh viên (nếu áp dụng).
    - Dữ liệu học phí cập nhật (học phí đã đóng, còn lại).

2. Luồng chính (Main Flow)

1. Người dùng (sinh viên hoặc admin) truy cập tính năng tính học phí qua API hoặc UI (ví dụ: GET /api/finance/tuition/programs/[id]/calculate hoặc trang tính học phí chương trình).
2. Hệ thống xác thực quyền truy cập và lấy dữ liệu đầu vào: program_id, year, tuition_group (Standard, Premium, Scholarship, Reduced), calculation_type (PER_CREDIT, FIXED, COURSE_BASED, MIXED), use_fallback (true/false).
3. Hệ thống ưu tiên kiểm tra học phí theo Program:
    - Tìm ProgramTuition theo program_id, year, tuition_group.
    - Nếu có, tính học phí dựa trên calculation_method:
        - FIXED: Sử dụng amount_vnd.

        - PER_CREDIT: total_credits (từ Program) × per_credit_fee + misc_fee.
        - COURSE_BASED: Tính từng học phần từ ProgramCourseMap và CourseTuition (per_credit_fee × credits, hoặc fixed_fee, theory_fee/practice_fee nếu áp dụng).
        - MIXED: Kết hợp các phương thức (ví dụ: FIXED cho một phần + PER_CREDIT cho phần còn lại).
4. Nếu không có ProgramTuition, fallback sang:
    - TuitionRule với scope = 'P' (program-specific).
    - Nếu vẫn không, fallback sang MajorTuition (dựa trên major_id từ Program) hoặc TuitionRule scope = 'M'.
    - Cuối cùng, TuitionRule scope = 'S' (system-wide).
5. Hệ thống tính thêm chi tiết:
    - Breakdown: Theo học phần (byCourse), khối (byBlock), nhóm (byGroup).
    - Đối với sinh viên cụ thể: Tích hợp StudentAcademicProgress (credits_earned, credits_remaining), Invoice (paid_tuition = sum(amount where status='paid')).
    - Học phí còn lại = total_tuition - paid_tuition.
    - Dự báo: Dựa trên credits_remaining, học phí dự kiến theo học kỳ/năm (giả sử tốc độ tích lũy tín chỉ trung bình).
6. Hệ thống lưu kết quả tính toán vào ProgramTuitionCalculation (cho chương trình) hoặc StudentTuitionSummary (cho sinh viên).
7. Hệ thống trả về kết quả (JSON response) hoặc hiển thị trên UI, bao gồm total_tuition, breakdown, source (PROGRAM_TUITION, MAJOR_TUITION, etc.).
8. Nếu là admin, cho phép phê duyệt và áp dụng (cập nhật status='active').
9. Kết thúc Use Case.

3. Luồng thay thế (Alternative Flows)

- **3.1 Tính học phí theo học phần cụ thể**:
    - Từ bước 3, nếu calculation_type = COURSE_BASED, hệ thống lặp qua ProgramCourseMap để tính từng course_id.
    - Phân biệt is_required (bắt buộc/tự chọn), type (theory, practice, project) để áp dụng phí khác nhau từ CourseTuition (theory_fee, practice_fee).
    - Tiếp tục từ bước 5.
- **3.2 Dự báo học phí cho sinh viên**:
    - Từ bước 5, nếu yêu cầu dự báo (query param includeForecast=true), hệ thống tính dựa trên enrollment_date, expected_graduation_date, credits_earned.
    - Dự báo theo học kỳ: (credits_remaining / credits_per_semester_avg) × per_semester_fee.
    - Theo năm: Tương tự, nhóm theo năm.
    - Tiếp tục từ bước 7.
- **3.3 Quản lý học phí (CRUD)**:
    - Nếu actor là admin, từ bước 1, chọn quản lý (ví dụ: POST /api/finance/tuition/programs để tạo ProgramTuition mới).
    - Hệ thống kiểm tra unique constraint (program_id, year, tuition_group).
    - Cập nhật effective_from/effective_to, status.
    - Tiếp tục từ bước 6 để tính lại nếu cần.

- **3.4 Báo cáo học phí**:
    - Từ bước 7, nếu yêu cầu báo cáo (GET /api/finance/tuition/reports/programs), hệ thống tổng hợp dữ liệu từ ProgramTuitionCalculation, StudentTuitionSummary.
    - Báo cáo theo chương trình (tổng học phí, so sánh chương trình), sinh viên (lịch sử thanh toán), học kỳ (tỷ lệ thanh toán).
    - Xuất file (PDF/Excel).

4. Luồng ngoại lệ (Exception Flows)

- **4.1 Không tìm thấy dữ liệu học phí**:
    - Từ bước 3, nếu không có ProgramTuition và không fallback (use_fallback=false), hoặc không có fallback nào, hệ thống trả lỗi: "No tuition data found for the program/major in the specified year."
    - Kết thúc Use Case với response {success: false, error: "NO_DATA"}.
- **4.2 Dữ liệu không hợp lệ**:
    - Từ bước 2, nếu program_id không tồn tại hoặc year ngoài effective_from/effective_to, hệ thống trả lỗi validation.
    - Ghi log và thông báo người dùng sửa input.
- **4.3 Lỗi tính toán (ví dụ: credits âm)**:
    - Từ bước 5, nếu credits_remaining < 0 hoặc total_tuition không tính được (dữ liệu thiếu), hệ thống fallback sang method default (PER_CREDIT) và ghi note: "Calculation fallback due to invalid data."
    - Thông báo admin kiểm tra dữ liệu.
- **4.4 Quyền truy cập bị từ chối**:
    - Từ bước 1, nếu người dùng không có quyền (sinh viên chỉ xem cá nhân, admin quản lý), hệ thống trả lỗi 403 Forbidden.

5. Quy tắc nghiệp vụ (Business Rules)

- Ưu tiên: ProgramTuition > TuitionRule (P) > MajorTuition > TuitionRule (M) > TuitionRule (S).
- Học phí tính theo VNĐ (amount_vnd), hỗ trợ misc_fee (phí khác).
- Thời gian hiệu lực: Chỉ tính nếu current_date trong effective_from/effective_to.
- Unique constraints: Đảm bảo không duplicate (ví dụ: program_id + year + tuition_group).
- Tích hợp Invoice: Chỉ tính status='paid' cho paid_tuition.
- Dự báo: Giả sử 15-20 credits/học kỳ trung bình, điều chỉnh dựa trên cohort_id.

6. Yêu cầu phi chức năng (Non-Functional Requirements)

- **Hiệu suất**: Tính toán < 2 giây cho CTĐT có 100 học phần.
- **Bảo mật**: Chỉ sinh viên xem dữ liệu cá nhân; admin xem tất cả. Sử dụng JWT authentication.
- **Khả dụng**: 99.9%, hỗ trợ caching cho breakdown.
- **Giao diện**: UI thân thiện (list view, form CRUD, charts báo cáo).

- **Tích hợp**: Kết nối database Prisma, schema academic/finance.

7. Use Case liên quan (Related Use Cases)

- Quản lý CTĐT (tạo Program, Course).
- Thanh toán học phí (tạo Invoice, cập nhật paid_at).
- Tiến độ học tập (cập nhật credits_earned).

8. Biểu đồ (Diagrams)

- **Use Case Diagram**: Actor (Sinh viên, Finance Manager, Admin) -> Use Case (Tính học phí hoàn thành CTĐT) -> Include (Quản lý học phí, Báo cáo học phí) -> Extend (Dự báo học phí).
- **Sequence Diagram**: User -> API -> Service (TuitionService) -> DB (ProgramTuition, Invoice) -> Return Response.

Use Case này dựa hoàn toàn vào phân tích cung cấp, đảm bảo ưu tiên Program và fallback hợp lý.